Reconciling modern machine learning and the bias-variance trade-off

Mikhail Belkin[1], Daniel Hsu[2], Siyuan Ma[1], and Soumik Mandal[1]

[1]*The Ohio State University, Columbus, OH*
[2]*Columbia University, New York, NY*

December 31, 2018

Abstract

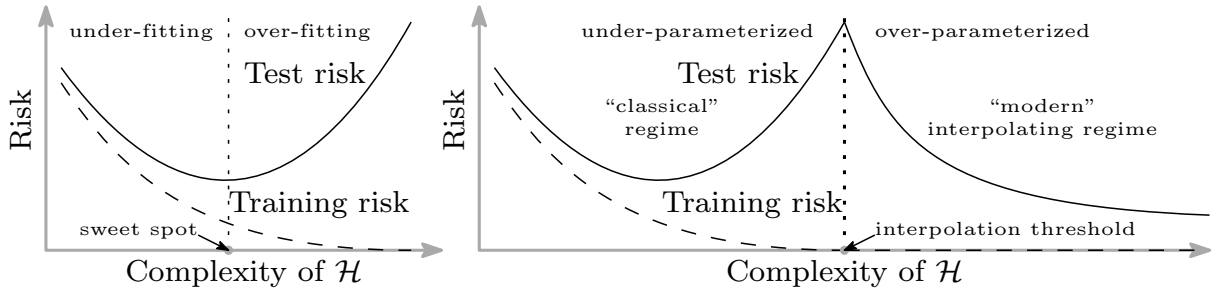
The question of generalization in machine learning—how algorithms are able to learn predictors from a training sample to make accurate predictions out-of-sample—is revisited in light of the recent breakthroughs in modern machine learning technology. The classical approach to understanding generalization is based on bias-variance trade-offs, where model complexity is carefully calibrated so that the fit on the training sample reflects performance out-of-sample. However, it is now common practice to fit highly complex models like deep neural networks to data with (nearly) zero training error, and yet these interpolating predictors are observed to have good out-of-sample accuracy even for noisy data. How can the classical understanding of generalization be reconciled with these observations from modern machine learning practice?

In this paper, we bridge the two regimes by exhibiting a new "double descent" risk curve that extends the traditional U-shaped bias-variance curve beyond the point of interpolation. Specifically, the curve shows that as soon as the model complexity is high enough to achieve interpolation on the training sample—a point that we call the "interpolation threshold"—the risk of suitably chosen interpolating predictors from these models can, in fact, be decreasing as the model complexity increases, often below the risk achieved using non-interpolating models. The double descent risk curve is demonstrated for a broad range of models, including neural networks and random forests, and a mechanism for producing this behavior is posited.

1 Introduction

Machine learning, which is central to many important applications in science and technology, focuses on the problem of prediction. Given a sample of training examples $(x_1, y_1), \ldots, (x_n, y_n)$ from $\mathbb{R}^d \times \mathbb{R}$, we learn a predictor $h_n \colon \mathbb{R}^d \to \mathbb{R}$ that is used to predict the label y of a new point x.

The predictor h_n is commonly chosen from some function class $\mathcal{H}$, such as linear functions or neural networks with a certain architecture, using *empirical risk minimization (ERM)* and its variants. In ERM, the predictor is taken to be a function $h \in \mathcal{H}$ that minimizes the *empirical (or training) risk* $\frac{1}{n}\sum_{i=1}^{n} \ell(h(x_i), y_i)$, where ℓ is a loss function such as the squared loss $\ell(y', y) = (y' - y)^2$ or zero-one loss $\ell(y', y) = \mathbb{1}_{\{y' \neq y\}}$. To study the out-of-sample (i.e., *generalization*) performance of h_n, we assume the training examples are sampled randomly from a probability distribution P over $\mathbb{R}^d \times \mathbb{R}$, and evaluate h_n on a new test example (x, y) is drawn independently from P. The analytical challenge stems from the mismatch between the goals of minimizing empirical risk (the explicit goal of ERM, optimization) and minimizing the *true (or test) risk* $\mathbb{E}_{(x,y)\sim P}[\ell(h(x), y)]$ (the goal of machine learning).



(a) U-shaped "bias-variance" risk curve

(b) "double descent" risk curve

Figure 1: Curves for training risk (dashed line) and test risk (solid line). (a) The classical *U-shaped risk curve* arising from the bias-variance trade-off. (b) The *double descent risk curve*, which incorporates the U-shaped risk curve (i.e., the "classical" regime) together with the observed behavior from using high complexity function classes (i.e., the "modern" interpolating regime), separated by the interpolation threshold. The predictors to the right of the interpolation threshold have zero training risk.

Most classical analyses of ERM are based on controlling the complexity of the function class $\mathcal{H}$ (sometimes called the *model complexity*) by managing the *bias-variance trade-off* (cf., [15]):

1. If $\mathcal{H}$ is too small, all predictors in $\mathcal{H}$ may *under-fit* the training data (i.e., have large empirical risk) and hence predict poorly on new data.

2. If $\mathcal{H}$ is too large, the empirical risk minimizer may *over-fit* spurious patterns in the training data that result in poor accuracy on new examples (i.e., small empirical risk but large true risk).

The classical analysis is concerned with finding the "sweet spot" for the function class complexity that balances these two concerns. The control of the function class complexity may be explicit, via the choice of $\mathcal{H}$ (e.g., picking the neural network architecture), or it may be implicit, using regularization (e.g., early stopping, complexity penalization). When a suitable balance is achieved, the performance of h_n on the training data is said to *generalize* to the population P. This is summarized in the classical U-shaped risk curve, shown in Figure 1(a).

Conventional wisdom in machine learning expounds the hazards of over-fitting. The textbook corollary of these hazards is that "a model with zero training error is overfit to the training data and will typically generalize poorly" [15].

However, practitioners routinely use modern machine learning methods to fit the training data perfectly or near-perfectly. For instance, highly complex neural networks and other non-linear predictors are often trained to have very low or even zero training risk. In spite of the high function class complexity and near-perfect fit to training data, these predictors often have excellent generalization performance—i.e., they give accurate predictions on new data. Indeed, this behavior has guided a best practice in deep learning for choosing the neural network architecture, specifically that the network architecture should be large enough to permit effortless interpolation of the training data [21]. Moreover, recent evidence indicates that neural networks and kernel machines trained to interpolate the training data obtain near-optimal test results even when the training data are corrupted with high levels of noise [28, 3].

In this work, we bridge the gap between classical statistical analyses and the modern practice of machine learning. We show that the classical U-shaped risk curve of the bias-variance trade-off, as well as the modern behavior where high function class complexity is compatible with good generalization behavior, can both be empirically witnessed with some important function classes including neural networks.

The main finding of this paper is summarized in the "double descent" risk curve shown in Figure 1(b). We observe that when function class complexity is below the "interpolation threshold", then learned predictors exhibit the classical U-shaped curve from Figure 1(a). (In this paper, function class complexity is identified with the number of parameters needed to specify a function within the class.) The bottom of the U is achieved at the sweet spot which balance between the fit to the training data and the susceptibility to over-fitting: to the left of the sweet spot, predictors are under-fit, and immediately to the right, predictors are over-fit. When we increase the function class complexity high enough (e.g., by increasing the number of features or the size of the neural network architecture), the learned predictors achieve (near) perfect fits to the training data—i.e., interpolation. Although the learned predictors obtained at the point of this *interpolation threshold* typically have high risk, we observe that increasing the function class complexity *beyond* this point leads to *decreasing* risk, in many cases going below the risk achieved at the sweet spot in the "classical" regime.

All of the learned predictors to the right of the interpolation threshold fit the training data perfectly. Therefore, the empirical risk is not a meaningful statistic by which to compare these predictors. So why should some—in particular, those from more complex functions classes—have lower (test) risk than others? The answer is that the complexity of the function class does not necessarily reflect how well the learned predictor matches the *inductive bias* appropriate for the problem at hand. For the learning problems we consider (both with real and synthetic data), the inductive bias that seems appropriate is a suitable function space norm that measures the regularity of a function. Choosing the smallest norm function that perfectly fits observed data is a form of Occam's razor: the simplest explanation compatible with the observations should be preferred (cf. [24, 5]). By considering larger function classes, which have higher complexity, we are able to find interpolating functions that have smaller norm and are thus "simpler". This explains why increasing the complexity of the function class can be beneficial.

Related ideas have been considered in the context of large margin classifiers [24, 1, 22], where a bigger function class $\mathcal{H}$ may permit the discovery of a classifier with a larger margin. However, the analyses used in these contexts do not predict the juxtaposition of the U-shaped risk curve with the descending risk curve beyond the interpolation threshold. Recently, there has been an emerging recognition that certain interpolating predictors (not based on ERM) can indeed be statistically optimal or near-optimal [2, 4], which is compatible with our empirical observations in the interpolating regime. A theory of interpolating ERM predictors still remains to be developed.

In the remainder of this article, we discuss empirical evidence for the double descent curve and exhibit the mechanism underlying its emergence.

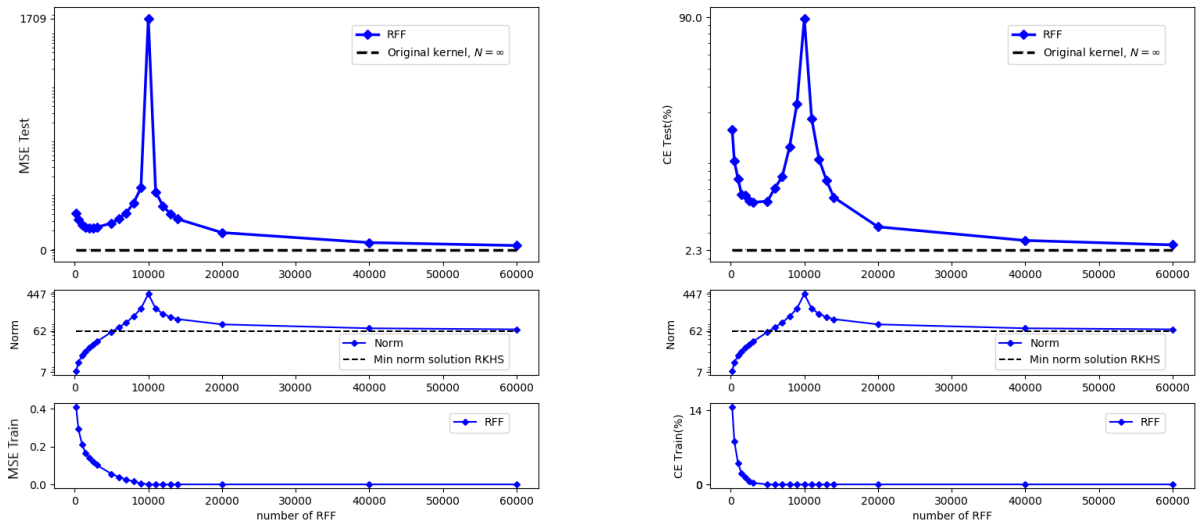
2 Neural networks

In this section, we discuss the double descent risk curve in the context of neural networks.

Random Fourier features. We first consider a popular class of non-linear parametric models called *Random Fourier Features* (*RFF*) [17], which can be viewed as a special class of two-layer neural networks with fixed weights in the first layer. The RFF model family $\mathcal{H}_N$ with N (complex-valued) parameters consists of functions $h\colon \mathbb{R}^d \to \mathbb{C}$ of the form

$$h(x) = \sum_{k=1}^{N} a_k \phi(x; v_k) \quad \text{where} \quad \phi(x; v) := e^{\sqrt{-1}\langle v_k, x\rangle},$$

and the vectors $v_1, \ldots, v_N$ are sampled independently from the standard normal distribution in $\mathbb{R}^d$. (We consider $\mathcal{H}_N$ as a class of real-valued functions with $2N$ real-valued parameters by taking real and imaginary



Squared loss Zero-one loss

Figure 2: Test risks, coefficient ℓ_2 norms, and training risks of the RFF model predictors $h_{n,N}$ learned on a subset of MNIST ($n = 10^4$, 10 classes).

parts separately.) Note that $\mathcal{H}_N$ is a *randomized* function class, but as $N \to \infty$, the function class becomes a closer and closer approximation to the Reproducing Kernel Hilbert Space (RKHS) corresponding to the Gaussian kernel, denoted by $\mathcal{H}_\infty$. While it is possible to directly use $\mathcal{H}_\infty$ by exploiting convex duality (e.g., as is done with kernel Support Vector Machines [6]), the random classes $\mathcal{H}_N$ are computationally attractive to use when the sample size n is large but the number of parameters N is comparably small. This computational advantage is not a focus of our paper, but we briefly revisit it in Section 4.

Our learning procedure using $\mathcal{H}_N$ is as follows. Given data $(x_1, y_1), \ldots, (x_n, y_n)$ from $\mathbb{R}^d \times \mathbb{R}$, we find the predictor $h_{n,N} \in \mathcal{H}_N$ via ERM with squared loss. That is, we minimize the empirical risk objective $\frac{1}{n}\sum_{i=1}^{n}(h(x_i) - y_i)^2$ over all functions $h \in \mathcal{H}_N$. When the minimizer is not unique (as is always the case when $N > n$), we choose the minimizer whose coefficients $(a_1, \ldots, a_N)$ have the minimum ℓ_2 norm. This choice of norm is intended as an approximation to the RKHS norm $\|h\|_{\mathcal{H}_\infty}$, which is generally difficult to compute for arbitrary functions in $\mathcal{H}_N$. For problems with multiple outputs (e.g., multi-class classification), we use functions with vector-valued outputs and sum of the squared losses for each output.

In Figure 2, we show the test risk of the predictors learned using $\mathcal{H}_N$ on a subset of the popular data set of handwritten digits called MNIST. The same figure also shows the ℓ_2 norm of the function coefficients, as well as the training risk. We see that for small values of N, the test risk shows the classical U-shaped curve consistent with the bias-variance trade-off, with a peak occuring at the interpolation threshold $N = n$. Some statistical analyses of RFF suggest choosing $N \propto \sqrt{n}\log n$ to obtain good test risk guarantees [19].

The interpolation regime connected with modern practice is shown to the right of the interpolation threshold, with $n \geq N$. The least complex model class that achieves interpolation (at $N = n$ random features) yields the least accurate predictor. (In fact, it has no predictive ability for classification.) But as the number of features increases beyond n, the accuracy improves dramatically, exceeding that of the predictor corresponding to the bottom of the U-shaped curve. The plot also shows that the predictor $h_{n,\infty}$ obtained from $\mathcal{H}_\infty$ (i.e., the kernel machine) out-performs the predictors from $\mathcal{H}_N$ for any finite N.

While the full mathematical and statistical details of this phenomenon remain to be investigated, we

provide some intuition for the underlying mechanism that leads to this double descent risk curve. When the number of random features is much smaller then the sample size, $N \ll n$, classical statistical arguments imply that the training risk is close to the test risk. Thus, for small N, adding more features can yield improvements in both the training and test risks. However, as the number of features approaches n (the interpolation threshold), features not present or only weakly present in the data are forced to fit the training data. This results in classical over-fitting as predicted by the bias-variance trade-off.

To the right of the interpolation threshold, the function classes are rich enough to achieve zero training risk. For the classes $\mathcal{H}_N$ that we consider, there is no guarantee that the smallest norm predictor consistent with training data (namely $h_{n,\infty}$, which is in $\mathcal{H}_\infty$) is contained in the class $\mathcal{H}_N$ for any finite N. But increasing N allows us to construct progressively better approximations to that smallest norm function. Thus we expect to have predictors with largest norm at the interpolation threshold and for the norm to decrease monotonically as N increases. This is what we observe in Figure 2, and indeed $h_{n,\infty}$ has better accuracy than all $h_{n,N}$ for any finite N. Favoring small norm interpolating predictors turns out to be a useful inductive bias on MNIST and many other data sets, a phenomenon that can be explained mathematically under some modeling assumptions that we describe in Appendix A, and that has been empirically observed [3].

In Appendix B.1, we present additional empirical evidence for the same double descent behavior using other data sets. In Appendix B.2 we demonstrate double descent for ReLU random feature models, a class of ReLU neural networks with a setting similar to that of RFF. Finally, in Appendix C, we describe a simple synthetic model, which can be regarded as a one-dimensional version of the RFF model, with which we observe the same behavior seen in Figure 2.

Neural networks and backpropagation. In general multilayer neural networks (beyond RFF or ReLU random feature models), a learning algorithm will tune all of the weights to fit the training data, typically using versions of Stochastic Gradient Descent (SGD), with backpropagation to compute partial derivatives. This flexibility increases the representational power of neural networks, but also makes ERM generally more difficult to implement. Nevertheless, as shown in Figure 3, we observe that increasing the number of parameters in fully-connected two-layer neural networks leads to a risk curve qualitatively similar to that observed with RFF models. That the test risk improves beyond the interpolation threshold is compatible with the conjectured "small norm" inductive biases of the common training algorithms for neural networks [14, 16].

The computational complexity of ERM with neural networks makes the double descent risk curve difficult to observe. Indeed, in the classical under-parametrized regime ($N \ll n$), the non-convexity of the ERM optimization problem causes the behavior of local search-based heuristics, like SGD, to be highly sensitive to their initialization. Thus, if only suboptimal solutions are found for the ERM optimization problems, increasing the size of a neural network architecture may not always lead to a corresponding decrease in the training risk. This suboptimal behavior can lead to high variability in both the training and test risks that masks the double descent curve.

It is common to use neural networks with extremely large number of parameters [10]. But to achieve interpolation for a single output (regression or two class classification) one expects to need at least as many parameters as there are data points. Moreover, if the prediction problem has more than one output (as in multi-class classification), then the number of parameters needed should be multiplied by the number of outputs. This is indeed the case empirically for neural networks shown in Figure 3. Thus, for instance, even networks with 10^8 parameters may not be large enough for interpolation on data sets as large as ImageNet [20]. In such cases, the classical regime of the U-shaped risk curve is more appropriate to understand generalization. For smaller data sets, these large neural networks would be firmly in the over-parameterized regime, and simply training to obtain zero training risk often results in good test performance [28].

Additional results with neural networks are given in Appendix B.3.

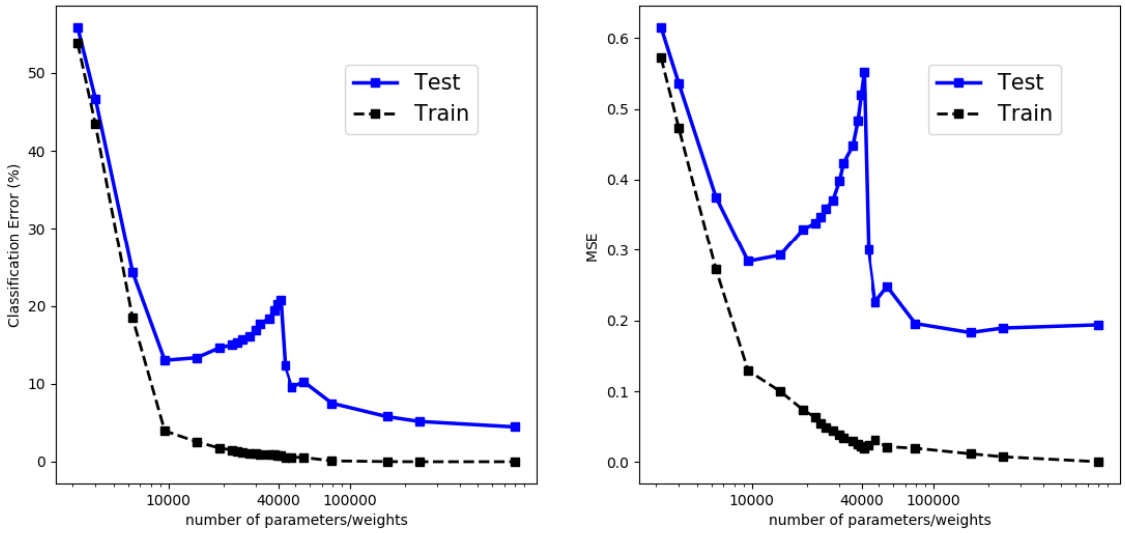


Figure 3: Training and test risks of fully connected neural networks with a single layer of H hidden units, learned on a subset of MNIST ($n = 4 \times 10^3$, $K = 10$ classes; left: zero-one loss; right: squared loss). The number of parameters is $(d + 1) \times H + (H + 1) \times K$. The interpolation threshold is observed at $n \times K$.

3 Decision trees and ensemble methods

Does the double descent risk curve manifest with other prediction methods besides neural networks? We give empirical evidence that the families of functions explored by boosting with decision trees and Random Forests also show similar generalization behavior as neural nets, both before and after the interpolation threshold.

AdaBoost and Random Forests have recently been investigated in the interpolation regime by Wyner et al. [27]. In particular, they give empirical evidence that, when AdaBoost and Random Forests are used with maximally large (interpolating) decision trees, the flexibility of the fitting methods yield interpolating predictors that are more robust to noise in the training data than the predictors produced by rigid, non-interpolating methods (e.g., AdaBoost or Random Forests with shallow trees). This in turn is said to yield better generalization. A crucial aspect of the predictors is that they are averages of these (near) interpolating trees (a property that is patent with Random Forests, but also observed for AdaBoost, as discussed by Wyner et al.): the averaging ensures that the resulting function is substantially smoother than any individual tree, which aligns with an inductive bias that is compatible with many real world problems.

We can understand these flexible fitting methods in the context of the double descent risk curve. Observe that the size of a decision tree (controlled by the number of leaves) is a natural way to parameterize the function class complexity: a tree with only two leaves corresponds to two-piecewise constant functions with axis-aligned boundary, while a tree with n leaves can interpolate n training examples. It is a classical observation that the U-shaped bias-variance trade-off curve manifests in many problems when the class complexity is considered this way [15]. (The interpolation threshold may be reached with fewer than n leaves in many cases, but n is clearly an upper bound.) To further enlarge the function class, we consider ensembles (averages) of several interpolating trees.[1] So, beyond the interpolation threshold, we use the number of such

[1]These trees are trained in the way proposed in Random Forest except without bootstrap re-sampling. This is similar to the PERT method of Cutler and Zhao [12].

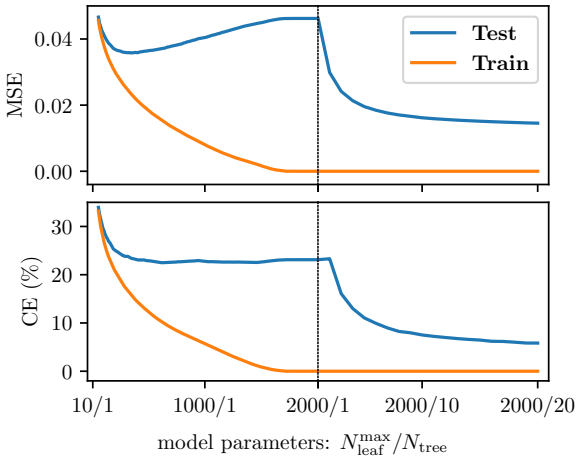


Figure 4: Double-descent risk curve of random forest trained for MNIST ($n = 10^4$)

trees to index the class complexity. When we view the risk curve as a function of class complexity defined in this hybrid fashion, we see the double descent curve appear just as with neural networks; see Figure 4. A similar phenomenon is observed using L_2-boosting with decision trees [13, 9], as reported in Appendix E.

4 Concluding thoughts

The double descent risk curve introduced in this paper extends the U-shaped curve predicted by the bias-variance trade-off to the observed behavior of complex models used in modern machine learning practice. The mechanism we suggest that underlies its emergence is based on common inductive biases, and hence can explain its appearance (and, we expect, ubiquity) in machine learning applications.

We conclude with some final remarks.

Historical absence. The double descent behavior may have been historically overlooked on account of several cultural and practical barriers. Observing the double descent curve requires a parametric family of spaces with functions of arbitrary complexity. The usual linear settings studied extensively in classical statistics typically assume a fixed set of features and hence fixed fitting capacity. Richer families of function classes are typically used in the context of non-parametric statistics, where smoothing and regularization are almost always employed [25]. Regularization, of all forms, can both prevent interpolation and change the effective capacity of the function class, thus attenuating or masking the interpolation peak.

The RFF models are a recently popular and flexible parametric family. However, these models were originally proposed as computationally favorable alternative to kernel methods. This computational advantage over traditional kernel methods holds only for $N \ll n$, and hence models at or beyond the interpolation threshold are typically not considered.

The situation with general neural networks, another parametric family of arbitrary complexity, is slightly different and more involved. Due to the non-convexity of the ERM optimization problem, solutions in the classical under-parametrized regime are highly sensitive to initialization. Moreover, as we observed, the peak at the interpolation threshold is often localized and easy to miss, leading to the impression that increasing the size of the network simply improves performance. Finally, in practice, training of neural networks is typically stopped as soon as (an estimate of) the test risk fails to improve. This early stopping has a regularizing effect that, as discussed above, makes it difficult to observe the interpolation peak.

Inductive bias. In this paper, we have dealt with several types of methods for choosing interpolating solutions. For Random Fourier features, solutions are constructed explicitly by minimum norm linear regression in the feature space. As the number of features tends to infinity they approach the minimum functional norm solution in the Reproducing Kernel Hilbert Space, a solution which maximizes functional smoothness subject to the interpolation constraints. For neural networks, the inductive bias owes to the specific training procedure used, which is typically SGD. When all but the final layer of the network are fixed (as in RFF models), SGD converges to the minimum norm solution. While the behavior of SGD for more general neural networks is not fully understood, there is significant empirical and some theoretical evidence (e.g., [14]) that a similar minimum norm inductive bias is present. Finally, yet another type of inductive bias is used in random forests. Averaging potentially non-smooth interpolating solutions leads to an interpolating solution with a higher degree of smoothness; see [7] for some analysis of infinite ensembles of trees.

Remarkably, for kernel machines all three methods lead to the same minimum norm solution. Indeed, the minimum norm interpolating classifier, $h_{n,\infty}$, can be obtained directly by explicit norm minimization (solving an explicit system of linear equations), through SGD or by averaging trajectories of Gaussian processes (computing the posterior mean [18]).

Optimization and practical considerations. In our experiments, appropriately chosen "modern" models usually outperform the optimal "classical" model on the test set. But another important practical advantage of over-parametrized models is in optimization. There is a growing understanding that larger models are "easy" to optimize in the sense that local methods, such as SGD, converge to global minima of the training risk in over-parameterized regimes (e.g., [23]). Thus, large interpolating models can have low test risk and be easy to optimize at the same time.

Outlook. We hope that the perspective developed in this work will lead to better understanding of common machine learning models, as well as new intuitions and algorithms for parameter and architecture selection. Our observations open new lines of theoretical inquiry to study computational, statistical, and mathematical properties of the classical and interpolating regimes in modern machine learning.

References

[1] Peter L. Bartlett. The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network. *IEEE Transactions on Information Theory*, 44 (2):525–536, 1998.

[2] Mikhail Belkin, Daniel Hsu, and Partha Mitra. Overfitting or perfect fitting? risk bounds for classification and regression rules that interpolate. In *Advances in Neural Information Processing Systems*, 2018.

[3] Mikhail Belkin, Siyuan Ma, and Soumik Mandal. To understand deep learning we need to understand kernel learning. In *Proceedings of the 35th International Conference on Machine Learning*, 2018.

[4] Mikhail Belkin, Alexander Rakhlin, and Alexandre B Tsybakov. Does data interpolation contradict statistical optimality? *arXiv preprint arXiv:1806.09471*, 2018.

[5] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K Warmuth. Occam's razor. *Information processing letters*, 24(6):377–380, 1987.

[6] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM, 1992.

[7] Leo Breiman. Some infinity theory for predictor ensembles. Technical report, Technical Report 579, Statistics Dept. UCB, 2000.

[8] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[9] Peter Bühlmann and Bin Yu. Boosting with the l_2 loss: regression and classification. *Journal of the American Statistical Association*, 98(462):324–339, 2003.

[10] Alfredo Canziani, Adam Paszke, and Eugenio Culurciello. An analysis of deep neural network models for practical applications. *arXiv preprint arXiv:1605.07678*, 2016.

[11] Youngmin Cho and Lawrence K. Saul. Kernel methods for deep learning. In *Advances in Neural Information Processing Systems 22*, pages 342–350. 2009.

[12] Adele Cutler and Guohua Zhao. Pert-perfect random tree ensembles. *Computing Science and Statistics*, 33:490–497, 2001.

[13] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, pages 1189–1232, 2001.

[14] Suriya Gunasekar, Blake E Woodworth, Srinadh Bhojanapalli, Behnam Neyshabur, and Nati Srebro. Implicit regularization in matrix factorization. In *Advances in Neural Information Processing Systems*, pages 6151–6159, 2017.

[15] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer, 2001.

[16] Yuanzhi Li, Tengyu Ma, and Hongyang Zhang. Algorithmic regularization in over-parameterized matrix sensing and neural networks with quadratic activations. In Sébastien Bubeck, Vianney Perchet, and Philippe Rigollet, editors, *Proceedings of the 31st Conference On Learning Theory*, volume 75 of *Proceedings of Machine Learning Research*, pages 2–47. PMLR, 06–09 Jul 2018.

[17] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems*, pages 1177–1184, 2008.

[18] Carl Edward Rasmussen. Gaussian processes in machine learning. In *Advanced Lectures on Machine Learning*, pages 63–71. Springer, 2004.

[19] Alessandro Rudi and Lorenzo Rosasco. Generalization properties of learning with random features. In *Advances in Neural Information Processing Systems*, pages 3215–3225, 2017.

[20] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3): 211–252, 2015. doi: 10.1007/s11263-015-0816-y.

[21] Ruslan Salakhutdinov. Deep learning tutorial at the Simons Institute, Berkeley, https://simons.berkeley.edu/talks/ruslan-salakhutdinov-01-26-2017-1, 2017. URL https://simons.berkeley.edu/talks/ruslan-salakhutdinov-01-26-2017-1.

[22] Robert E. Schapire, Yoav Freund, Peter Bartlett, and Wee Sun Lee. Boosting the margin: a new explanation for the effectiveness of voting methods. *Ann. Statist.*, 26(5):1651–1686, 1998.

[23] Mahdi Soltanolkotabi, Adel Javanmard, and Jason D Lee. Theoretical insights into the optimization landscape of over-parameterized shallow neural networks. *IEEE Transactions on Information Theory*, 2018.

[24] Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1995. ISBN 0-387-94559-8.

[25] Larry Wasserman. *All of Nonparametric Statistics*. Springer, 2006.

[26] Holger Wendland. *Scattered Data Approximation*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, 2004. doi: 10.1017/CBO9780511617539.

[27] Abraham J Wyner, Matthew Olson, Justin Bleich, and David Mease. Explaining the success of adaboost and random forests as interpolating classifiers. *Journal of Machine Learning Research*, 18(48): 1–33, 2017.

[28] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations*, 2017.

A Approximation theorem

Suppose the training data $(x_1, y_1), \ldots, (x_n, y_n)$ are sampled independently by drawing x_i uniformly from a compact domain in $\mathbb{R}^d$, and assigning the label $y_i = h^*(x_i)$ using a target function $h^* \in \mathcal{H}_\infty$. Let $h \in \mathcal{H}_\infty$ be another hypothesis that interpolates the training data $(x_1, y_1), \ldots, (x_n, y_n)$. The following theorem bounds the error of h in approximating h^*.

Theorem 1. *Fix any $h^* \in \mathcal{H}_\infty$. Let $(x_1, y_1), \ldots, (x_n, y_n)$ be independent and identically distributed random variables, where x_i is drawn uniformly at random from a compact cube[2] $\Omega \subset \mathbb{R}^d$, and $y_i = h^*(x_i)$ for all i. There exists absolute constants $A, B > 0$ such that, for any interpolating $h \in \mathcal{H}_\infty$ (i.e., $h(x_i) = y_i$ for all i), so that with high probability*

$$\sup_{x \in \Omega} |h(x) - h^*(x)| < A e^{-B(n/\log n)^{1/d}} \left(\|h^*\|_{\mathcal{H}_\infty} + \|h\|_{\mathcal{H}_\infty} \right).$$

Proof sketch. Recall that *the fill* κ_n of the set of points $x_1, \ldots, x_n$ in Ω is a measure of how well these points cover Ω: $\kappa_n = \max_{x \in \Omega} \min_{x_j \in \{x_1, \ldots, x_n\}} \|x - x_j\|$. It is easy to verify (e.g., by taking an appropriate grid partition of the cube Ω and applying the union bound) that with high probability $\kappa_n = O(n/\log n)^{-1/d}$.

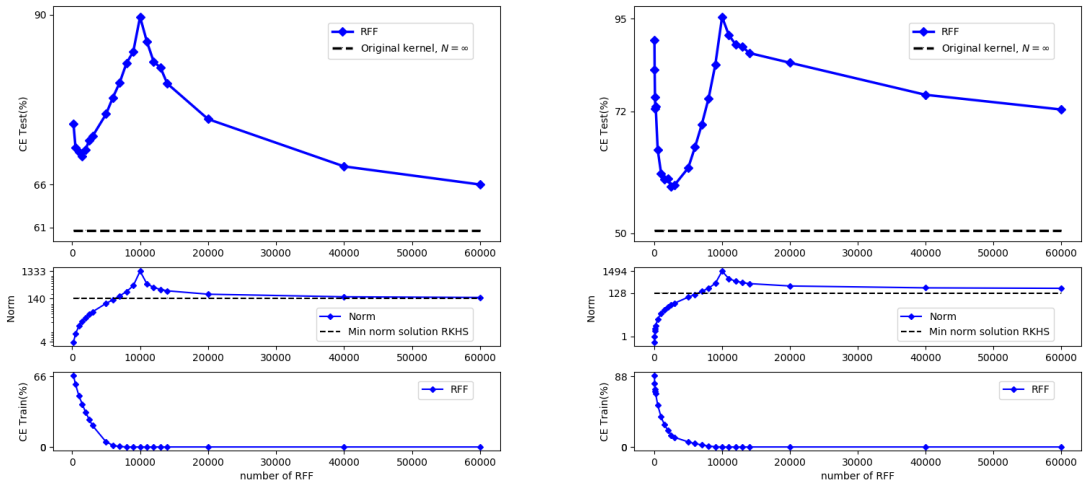
Consider now a function $f(x) := h(x) - h^*(x)$. We observe that $f(x_i) = 0$ and by the triangle inequality $\|f\|_{\mathcal{H}_\infty} \leq \|h^*\|_{\mathcal{H}_\infty} + \|h\|_{\mathcal{H}_\infty}$. Applying Theorem 11.22 in [26] to f yields the result. □

The minimum norm interpolating function $h_{n,\infty}$ has norm no larger than that of h^* (by definition) and hence achieves the smallest bound in Theorem 1. While these bounds apply only in the noiseless setting, they provide a justification for the inductive bias based on choosing a solution with a small norm. Indeed, there is significant empirical evidence that minimum norm interpolating solutions generalize well on a variety of datasets, even in the presence of large amounts of label noise [3].

[2] Same argument can be used for more general domains and probability distributions.

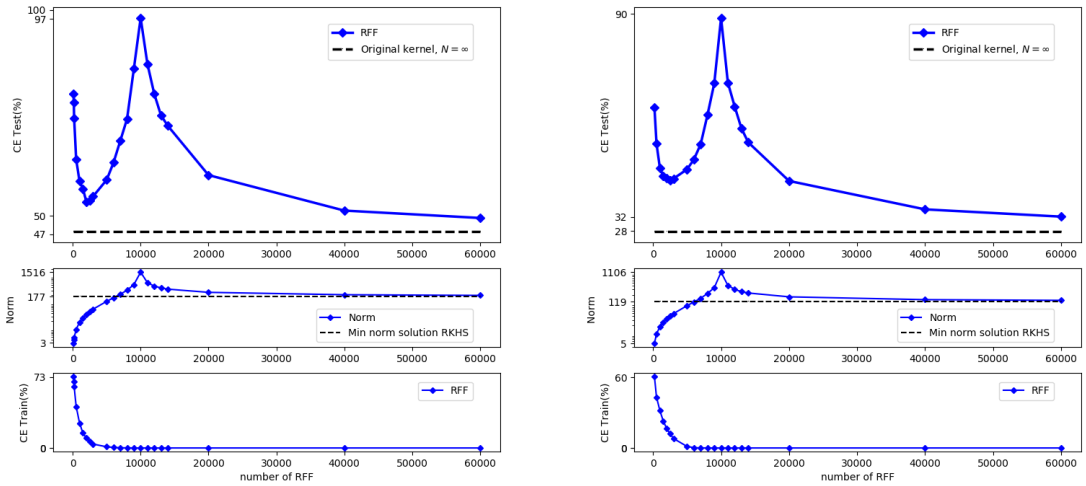
B Additional results with RFF models and neural networks

B.1 RFF models



(a) CIFAR-10 (b) 20Newsgroup

Figure 5: Test risks, coefficient ℓ_2 norms, and training risks of the RFF model predictors $h_{n,N}$ learned on subsets of CIFAR-10 and 20Newsgroups data ($n = 10^4$, zero-one loss).

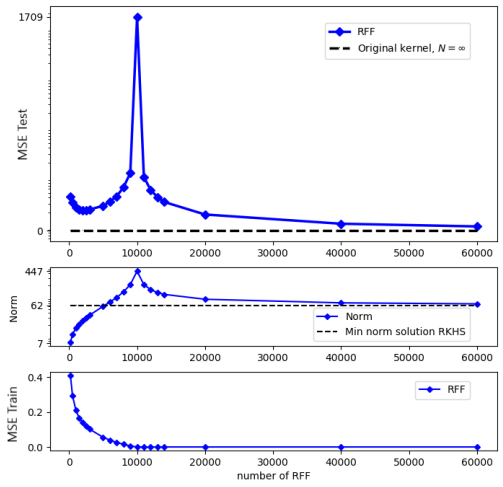


(a) TIMIT (b) SVHN

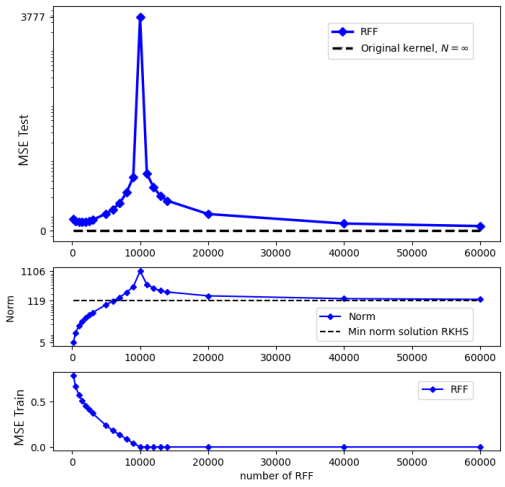
Figure 6: Test risks, coefficient ℓ_2 norms, and training risks of the RFF model predictors $h_{n,N}$ learned on subsets of TIMIT and SVHN data ($n = 10^4$, zero-one loss).

We provide additional experimental results for several real-world datasets. Figure 5 illustrates double descent behavior for CIFAR-10 and 20Newsgroup. Figure 6 shows TIMIT and SVHN.

While curves for squared loss are slightly different from the curves for zero-one loss, we observe the same kind of double descent behavior. The squared loss results for MNIST and SVHN are shown in Figure 7.



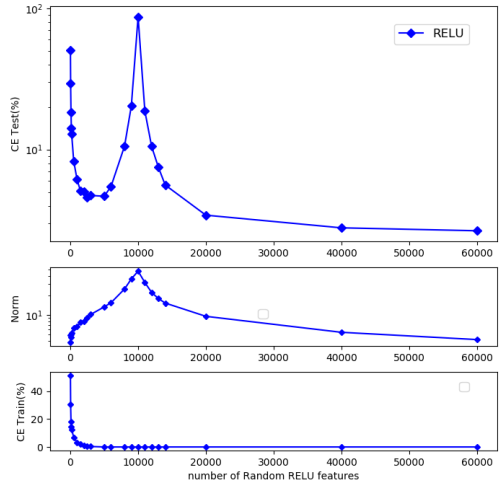
(a) MNIST



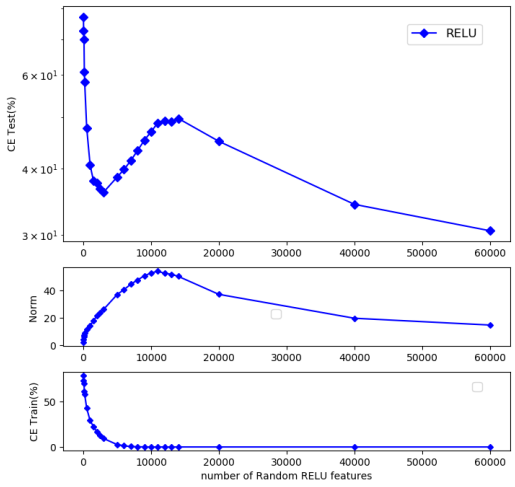
(b) SVHN

Figure 7: Test risks, coefficient ℓ_2 norms, and training risks of the RFF model predictors $h_{n,N}$ learned on subsets of MNIST and SVHN data ($n = 10^4$, squared loss).

B.2 Random ReLU Features



(a) MNIST



(b) SVHN

Figure 8: Test risks, coefficient ℓ_2 norms, and training risks of the Random ReLU Features model predictors $h_{n,N}$ learned on subsets of MNIST and SVHN data ($n = 10^4$, zero-one loss).

Below we provide some experimental results demonstrating double descent for models based on random ReLU feature netoworks [11]. That model is a special type of 2-layer neural networks with the popular ReLU transfer function and fixed first layer weights.

Specifically, the ReLU model family $\mathcal{H}_N$ with N parameters consists of functions $h\colon \mathbb{R}^d \to \mathbb{R}$ of the form

$$h(x) = \sum_{k=1}^{N} a_k \phi(x; v_k) \quad \text{where} \quad \phi(x; v) := \max(\langle v_k, x \rangle, 0)$$

Similarly to RFF, the vectors $v_1, \ldots, v_N$ are sampled independently from the standard normal distribution in $\mathbb{R}^d$ and the coefficients a_k are learned using linear regression. Figure 8 illustrates CE with Random ReLU features for MNIST and SVHN data. Regularization of 10^{-5} was added in SVHN experiments to ensure numerical stability near the interpolation threshold and leading to an "attenuated" interpolation peak. We observe that the resulting risk curves and the norm curves are very similar to those observed for RFF.

B.3 Neural networks

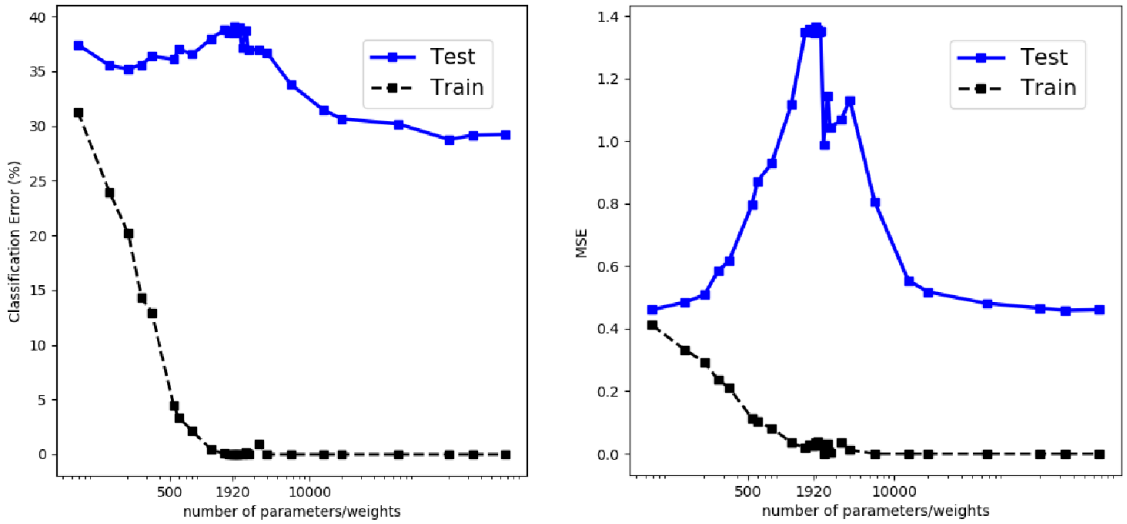


Figure 9: Training and test risks of fully connected neural networks with a single layer of H hidden units, learned on a subset of CIFAR-10 ($n = 960$, $K = 2$ classes; left: zero-one loss; right: squared loss). The interpolation threshold is observed around $N = n \times K$.

As mentioned in Section 2, the ERM optimization problem with neural networks is generally more difficult to solve computationally than the corresponding problem with RFF models. The learning algorithms we use to fit neural networks employ stochastic gradient methods, which are sensitive to their initialization. To mitigate this sensitivity, in the regime where $N < n$, we use parameters obtained by training smaller neural network as initialization when training the larger networks (weight reuse). This procedure ensures decreasing training loss as the number of parameters increases. We switch to using standard (random) initialization in the over-parametrized regimes ($N > n$), and typically have no difficulty obtaining near-zero training risk. The result for MNIST were shown in Figure 3. Figure 9 shows results of fully connected neural network for CIFAR-10 (2 class).

Results for MNIST without weight re-use are shown in Figure 10. In that setting, all models are randomly initialized for every number of parameters, instead of using the weight-reuse scheme and the average is plotted. While the variance is significantly larger, and the training loss is not monotonically decreasing, the double descent behavior is still clearly discernible.

C Synthetic model

We now discuss the nature of the double descent risk curve in the context of a simple synthetic model, which can be viewed as a version of RFF for functions on the one-dimensional circle. Consider the class $\mathcal{H}$

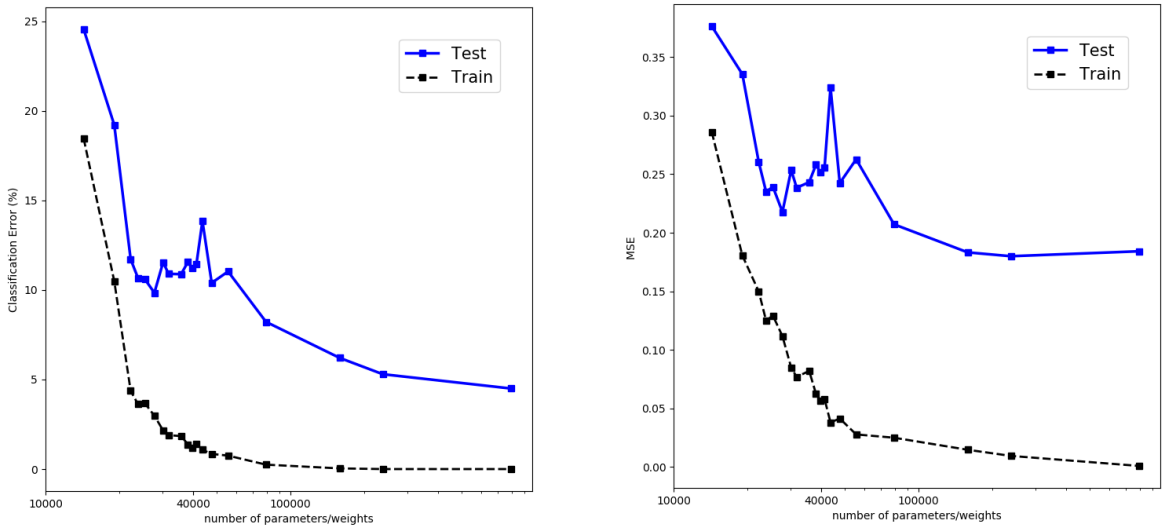


Figure 10: Training and test risks of fully connected neural networks with a single layer of H hidden units, learned on a subset of MNIST ($n = 4000$, $K = 10$ classes; left: zero-one loss; right: squared loss). The interpolation threshold is observed around $N = n \times K$.

of periodic complex-valued functions on the interval $[0, 2\pi]$, and let

$$e_k(x) := \exp(\sqrt{-1}(k-1)x)$$

for positive integers k. Fix a probability distribution $p = (p_1, p_2, \ldots)$ on the positive integers. For each integer N, we generate a random function class $\mathcal{H}_N$ by (i) sampling independently from p until N distinct indices $k_1, \ldots, k_N$ are chosen, and then (ii) let $\mathcal{H}_N$ be the linear span of $e_{k_1}, \ldots, e_{k_N}$. Here, N is the number of parameters to specify a function in $\mathcal{H}_N$ and also reflects the complexity of $\mathcal{H}_N$.

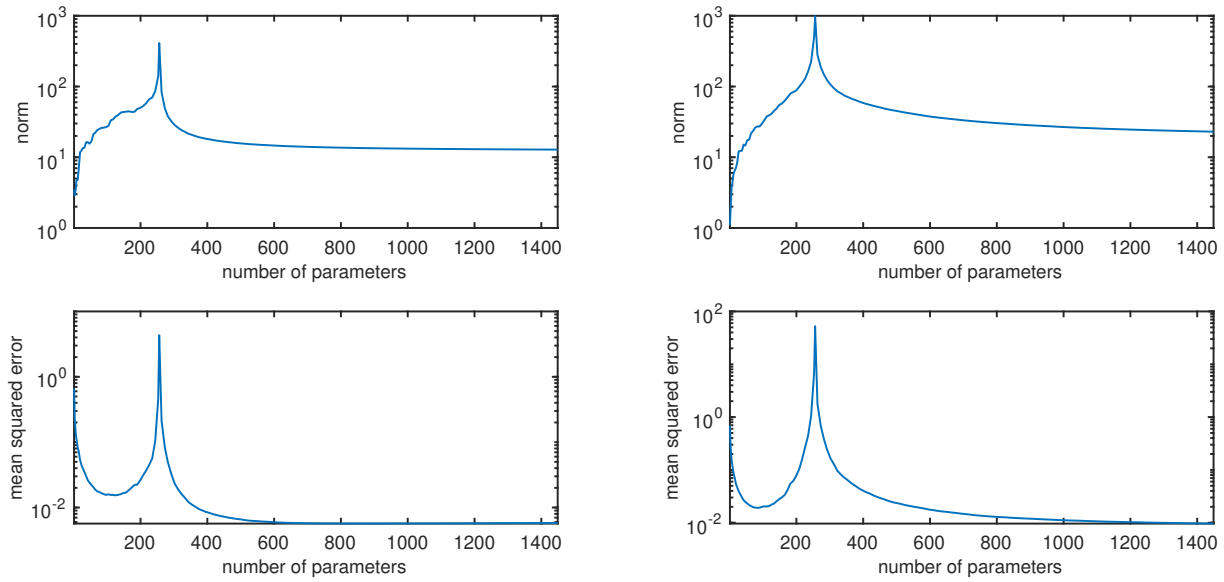
We generate data from the following model:

$$y_i = h^*(x_i) + \varepsilon_i$$

where the target function $h^* = \sum_k \alpha_k^* e_k$ is in the span of the e_k, and $\varepsilon_1, \ldots, \varepsilon_n$ are independent zero-mean normal random variables with variance σ^2. The $x_1, \ldots, x_n$ themselves are drawn uniformly at random from $\{2\pi j/M : j = 0, \ldots, M-1\}$ for $M := 4096$. We also let $\alpha_k^* := p_k$ for all k. The signal-to-noise ratio (SNR) is $\mathbb{E}[h^*(x_i)^2]/\sigma^2$.

Given data $(x_1, y_1), \ldots, (x_n, y_n) \in [0, 2\pi] \times \mathbb{R}$, we learn a function from the function class $\mathcal{H}_N$ using empirical risk minimization, which is equivalent to ordinary least squares over an N-dimensional space. Interpolation is achieved when $N \geq n$, so in this regime, we choose the interpolating function $h = \sum_{j=1}^N \alpha_{k_j} e_{k_j}$ of smallest (squared) norm $\|h\|_{\mathcal{H}}^2 = \sum_k \alpha_k^2/p_k$.

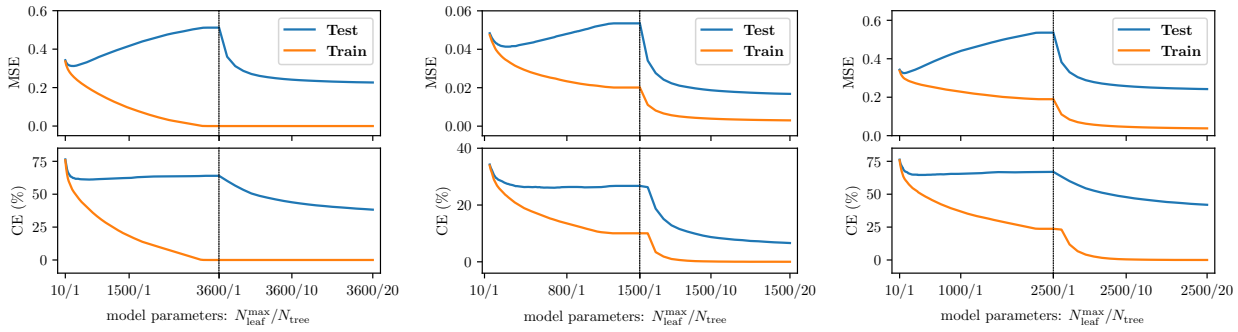
Our simulations were carried out for a variety of sample sizes ($n \in \{64, 128, 256, 512, 1024, 2048\}$) and are all repeated independently 20 times; our plots show averages over the 20 trials. The results confirm our hypothesized double descent risk curve, as shown in Figure 11 for $n = 256$; the results are similar for other n. The peak occurs at $N = n$, and the right endpoint of the curve is lower than the bottom of the U curve. The norm of the learned function also peaks at $N = n$ and decreases for $N > n$.



SNR = ∞ SNR = 20

Figure 11: Results from the synthetic model at SNR = ∞ and SNR = 20. Top: norm of learned function $\|h\|_{\mathcal{H}_\infty}$. Bottom: excess test risk under squared loss of learned function.

D Additional results with Random Forest



(a) SVHN without bootstrap re-sampling (b) MNIST with bootstrap re-sampling (c) SVHN with bootstrap re-sampling

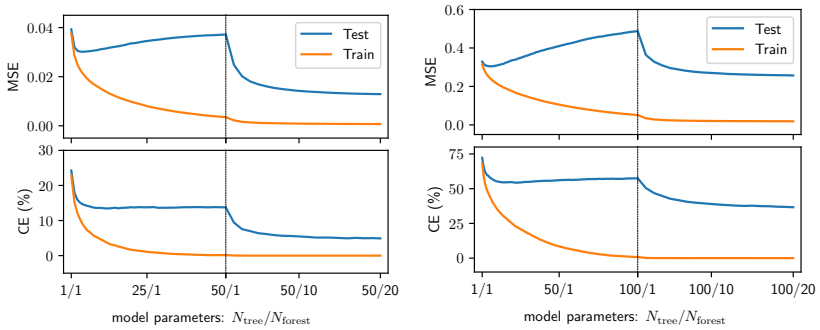
Figure 12: Double-descent risk curve for random forests trained with and without bootstrap re-sampling ($n = 10^4$). Top: squared loss. Bottom: zero-one loss.

We train standard random forests introduced in [8] for regression problems[3]. When splitting a node, we randomly select a subset of features whose number is the square root of the number of the total features, a setting which is widely used in mainstream implementations of random forest. We control the complexity of the forest by setting the number of trees (N_{tree}) and limiting the maximum number of leaves in each tree ($N_{\text{leaf}}^{\max}$). We put minimum constraints on the growth of each tree: there is no limit for the tree depth and we split each tree node whenever it is possible.

[3]Here we treat multi-class classification problem as regression problem with multiple outputs for consistency.

To interpolate the training data, we disable the bootstrap re-sampling for results in Figure 4 and Figure 12(a), which is similar to the PERT method in [12]. We see clear double decent risk curve (with both squared loss and zero-one loss) as we increase the complexity of the forest (although the U-shaped curve is less apparent with zero-one loss). In Figure 12(b) and Figure 12(c), we run the same experiments with bootstrap re-sampling enabled, which show similar double decent risk curves.

E Results with L_2-boosting



(a) MNIST with low shrinkage (b) SVHN with low shrinkage

Figure 13: Double-descent risk curve for L_2-boosting trees ($n = 10^4$). Top: squared loss. Bottom: zero-one loss.

We now show double descent risk curve for L_2-boosting (random) trees introduced in [13]. When splitting a node in a tree, we randomly select a subset of features whose number is the square root of the number of the total features. We constrain each tree to have a small number of leaves (no more than 10). As the number of trees increases, the boosted trees gradually interpolate the training data and form a forest. To quickly reach interpolation, we adopt low shrinkage (with parameter value 0.85) for gradient boosting. To go beyond the interpolation threshold, we average the predictions of several such forests which are randomly constructed and trained with exactly same hyper-parameters. The complexity of our model is hence controlled by the number of forests (N_{forest}) and the number of trees (N_{tree}) in each forest.

Figure 13(a) and Figure 13(b) show the change of train and test risk as the model complexity increases. We see the double descent risk curve for both squared loss and zero-one loss. We also observe strong overfitting under squared loss before the interpolation threshold. For similar experiments with high shrinkage (parameter value 0.1), the double descent risk curve becomes less apparent due to the regularization effect of high shrinkage [15].